Object Oriented Programming

FALL 2021 Solutions

1.A (i)

```
public interface P3 extends P1, P2 {

}
```

1.A (ii)

```
public class Main implements P3 {
	public static void main(String[] args) {
		Main var = new Main();
		var.k2();
	}

	public double h2(int x) {
		return x *= 100;
	}

	public String k1() {
		return "Hello, Java! ";
	}

	public void k2() {
		System.out.println(k1() + h2(5));
	}
}
```

1.B

```
public class Fall{
	static int x;
	final int y = 213;

	public static void main(String[] args) {
		x = 50;
	}
}
```

1.C

```
interface Person {
	void introduce();
}

public class Main{
	public static void main(String[] args) {
		Person engineer = new Person() {
			@Override
```

```
                public void introduce() {
                        System.out.println("Hello, I'm an Engineer");
                }
        };

        Person doctor = new Person() {
                @Override
                public void introduce() {
                        System.out.println("Hello, I'm a Doctor");
                }
        };

        engineer.introduce();
        doctor.introduce();
    }
}
```

2.A

```
public class ExceptionTest {
    public static void main(String[] args) {
        calculator obj = new calculator();
        try {
        obj.divide();
        obj.display_namelength();
        }
        catch(ArithmeticException e) {
                System.out.println("Exception handled successfully");
        }
        catch(NullPointerException e) {
                System.out.println("Exception handled successfully");
        }
    }
}
```

2.B

```
class Main {
    public static void main(String[] args)
    {
        try {
                int acc[] = { 100, 101, 102, 103, 104, 105 };
                double balance[] = { 2000, 1500, 900, 1560, 1765.50 };
                System.out.println("Account. No\t" + "Balance\t");
                for (int i = 0; i < 5; i++) {
                        System.out.println(acc[i] + "\t\t" + balance[i] + "\t");
                        if (balance[i] < 1000) {
                                // Task 1: Throw a new user-defined exception class
named "MinimumBalanaceException" here in this block.
                                throw new MinimumBalanaceException("Insufficient
balance");
                        }
                }
        }
        // Task 2: Write the catch block here. The catch block will catch the
exception generated by the custom user-defined class named "MinimumBalanaceException."
        catch(MinimumBalanaceException e) {
                System.out.println(e);
        }
    }
```

```
}
//Task 3: Write your MinimumBalanaceException class here. It will print the message
"Balance is low now."
class MinimumBalanaceException extends Exception {
	public MinimumBalanaceException(String s) {
		super(s);
	}
	@Override
	public String toString() {
		return "Balance is low now";
	}
}
```

3.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class Main {
	public static void main(String[] args) throws NumberFormatException, IOException
{
		final String inputFile = "//path";
		final String outputFile = "//path";

		FileReader fr = new FileReader(inputFile);
		BufferedReader br = new BufferedReader(fr);

		ArrayList<Integer> nums = new ArrayList<>();
		String buff;

		// Read numbers
		while(br.ready()) {
			buff = br.readLine();
			nums.add(Integer.parseInt(buff));
		}

		// Write the sums
		File f = new File(outputFile);
		if(!f.exists()) {
			f.createNewFile();
		}

		PrintWriter pw = new PrintWriter(outputFile);
		for(int i = 0; i < nums.size() - 1; i+=2) {
			int sum = nums.get(i) + nums.get(i + 1);
			pw.println("Line " + (i + 1) + ": " + sum);
		}

		pw.close();
		br.close();
		fr.close();
	}
}
```

4.

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Swapping App");
        frame.setSize(275, 120);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);

        JTextField tf1 = new JTextField(10);
        JTextField tf2 = new JTextField(10);
        JButton button = new JButton("Swap");
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String text = tf1.getText();
                tf1.setText(tf2.getText());
                tf2.setText(text);
            }
        });

        JPanel panel = new JPanel();
        JPanel textPanel = new JPanel(new FlowLayout());

        textPanel.add(tf1);
        textPanel.add(tf2);

        panel.add(textPanel, BorderLayout.NORTH);
        panel.add(button, BorderLayout.SOUTH);


        frame.setContentPane(panel);
        frame.setVisible(true);
    }
}
```

5.

```java
import java.util.ArrayList;
import java.util.Comparator;

class Student {
      String name;
      int id;
      double cgpa;

      public Student(String name, int id, double cgpa) {
            this.name = name;
            this.id = id;
            this.cgpa = cgpa;
      }

      public Student(String name) {
            this.name = name;
            this.id = -1;
            this.cgpa = -1;
      }
```

```
}

public class Main {
	public static void main(String[] args) {
		ArrayList<Student> list = new ArrayList<>() {
			@Override
			public boolean contains(Object o) {
				if(o instanceof String) {
					for(int i = 0; i < size(); i++) {
						if(((Student)get(i)).name.equals(o)) {
							return true;
						}
					}
				}
				return false;
			}
		};


		list.add(new Student("Wizard", 3, 3.88));
		list.add(new Student("Peter", 2, 3.5));
		list.add(new Student("Wanda", 1, 3.88));
		list.add(new Student("Thanos", 41, 3.9));
		list.add(new Student("Yelena", 7, 3.75));
		list.add(new Student("Thor", 15, 3.89));

		System.out.println("Contains \"Peter\"? " + list.contains("Peter"));

		list.sort(new Comparator<Student>() {
			@Override
			public int compare(Student s1, Student s2) {
				if(s1.cgpa > s2.cgpa) return 1;
				else if(s2.cgpa > s1.cgpa) return -1;
				else return 0;
			}});
	}
}
```

Alternative 5.A

```
import java.util.Scanner;

class MyThread extends Thread {
    int value;
    Scanner sn;
    public MyThread() {
        sn = new Scanner(System.in);
    }

    @Override
    public void run() {
        while(true) {
            value = sn.nextInt();
            if(value > 0) {
                System.out.println(value);
            }
        }
    }
}
public class Main
{
```

```
	public static void main(String[] args) {
		MyThread tr = new MyThread();
		tr.start();
	}
}
```

Alternative 5.B

```
import java.util.Scanner;

public class Main
{
	public static void main(String[] args) {
		Thread tr = new Thread(new EvenFilter());
		tr.start();
	}
}


class EvenFilter implements Runnable {
    int value;
    Scanner sn;
    public EvenFilter() {
        sn = new Scanner(System.in);
    }

    @Override
    public void run() {
        while(true) {
            value = sn.nextInt();
            if(value % 2 != 0) {
                System.out.println(value);
            }
        }
    }
}
```